git (altran)

devops, cursos, git

Ponente: **Daniel Sanchez**

conceptos

- HEAD: puntero al commit de trabajo

config

- `--system` : /etc/gitconfig
  - `git config --system color.status auto`
- `--global` : ~/.gitconfig
- `--local`: .git/config → por cada repositorio

valores

- `core.symlinks` : a **true** respeta los symlinks, a **false** (default) los trata como ficheros

commit

- `git commit –amend` : sobrescribe el último commit, editando mensaje → crea un nuevo commit con los cambios
  - si nos hemos dejado alguna cosa por añadir/modificar o hemos puesto un mensaje poco significativo o erróneo

checkout

- `git checkout -` : moverte a la rama anterior (de la que venías)

remote

- *upstreams* : remoto por defecto, configurado/ble en cada rama, donde sincronizará
  - `git push -u <remoto> <rama>` : define el *upstream* de la rama (al tiempo que está enviando). Al hacer un `git pull` no será necesario especificar

branch (ramas)

listar

- `git branch` : muestra ramas locales
  - `-v` : verboso
  - `-v –merged` : ramas mergeadas

    - `-v –no-merged` : ramas no mergeadas
    - `-a` : locales + remotas
    - `-avv` : locales + remotas + upstreams
    - `git branch <rama> -u <origin>/<rama>` : establece el upstream

crear

- `git branch <rama>` : crea rama desde la actual
    - `git branch <rama> <rama_madre>` crea rama desde rama_madre
- `git checkout <rama>` : nos movemos a rama
    - `git checkout -b <rama>` : creamos rama a partir de la actual y nos movemos a ella
    - `git checkout -b <rama> <rama_madre>`

sincronizar

- `git push <origin> <rama>` : envía rama al remoto <origin>
    - `-u` o `--set-upstream` : y establece el upstream
- `git pull` : descarga los cambios de la rama actual
    - `--all` : de todas las ramas
    - = git fetch + git merge @{u}
    - `--rebase` : reescribe la historia si esta ha sido reescritura en el remote. Puede evitar problemas de historias no coincidentes (DPS[1] lo usa por defecto)

borrar

borrar la 'rama' en local si está merged

git branch -d 'rama'

fuerza el borrado de la 'rama' local

git branch -D 'rama'

borra 'rama' en 'origin'

git push –delete [-d] 'origin' 'rama'

borra 'rama' en 'origin' (OJO 2 puntos)

git push 'origin' :'rama'

borra ramas borradas en 'origin'

git remote prune 'origin'

stash (limbo)

añadir

- `git stash` : guarda ficheros de *staged area*
  - `git stash save [-u | –include-untracked] «<message>»` : le da un nombre a la sesión guardada en el stash
  - `-u` : incluye los ficheros untracked
- `git stash [-u | –include-untracked] [-k|–keep-index]` : guarda en el stash los ficheros del *working copy*

listar

- `git stash list` : lista sesiones en el stash
- `git stash show stash@{indice}` : muestra los ficheros del stash

recuperar

- `git stash pop` : aplica los cambios en los ficheros guardados anteriormente y los saca del stash
  - `git stash apply stash@{indice}` : aplica los cambios en los ficheros del stash@{indice} (si hay más de uno)
  - `git stash pop stash@{indice}` : idem anterior pero elimina la sesión stash
- `git checkout stash@{indice} <fichero>` : recupera del stash el fichero indicado

borrar

- `git stash drop` : borra el primer elemento de la pila
- `git stash drop stash@{indice}` : borra la sesión indicada

> **borra todas las sesiones stash**
>
> git stash clear

tags

dos tipos:

- lightweight: ligeros, hace referencia a un commit
- annotated: añade más información, como autor, mensaje y fecha

añadir

- `git tag <tag>` : crea tag ligero en el HEAD
  - `git tag <tag> <commit>` : crea tag ligero del commit concreto
- `git tag -a <tag> -m «mensaje»` : crea tag anotado en el HEAD
  - `git tag -a <tag> <commit> -m «mensaje»` : idem en el commit indicado

listar / usar

- `git tag` : muestra lista de tags
- `git show <tag>` : información del tag
- `git checkout <tag>` : se usa como referencia para movernos al commit asociado

sincronizar

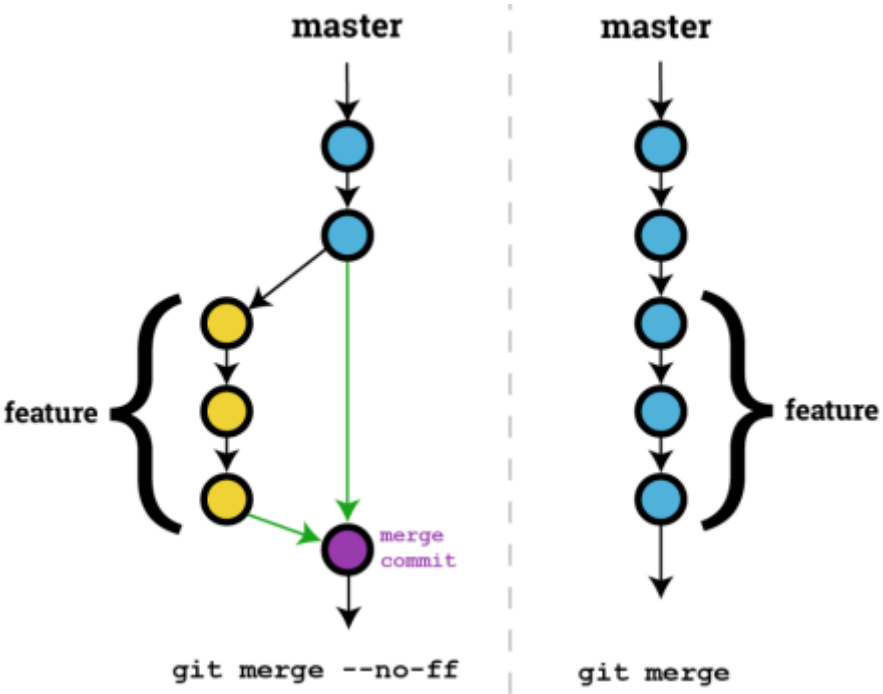
- `git push <remote> refs/tags/<tag>` : sube el tag al repositorio
- `git push –tags` : sube todos los tags
- `git pull –tags` : descarga todos los tags

merge

fusión a 3 bandas entre los dos últimos commits de las dos ramas y el ancestro común

2 maneras de mostrar/trabajar:

- `no-ff` → no fastforward : nanera de trabajar por defecto de la mayoría de los repositorios, se muestran las ramas creadas con sus propios commits
- `ff` → fastforward : se integra en una línea, sin mostrar las particularidades de una rama que ha sido mergeada



en ambos casos, al hacer el *merge* se trae todo el historial de esa rama

mergeando

- `git merge <rama>` : merge de la rama actual con <rama> en formato fast-forward
  - `--no-ff` : en formato no-fastforward
- `git merge –squash <rama>` : importa los cambios de la rama y los deja en *staged area*

cherry-pick

permite importar un commit (de otra rama) a tu rama sin traer el histórico (solo el commit en sí)

- `git cherry-pick <commit>` : trae el commit indicado a la rama actual
  - `-e` : edita el mensaje de commit
  - `<commit>..<commit>` : trae el rango de commits sucesivos indicados
    - `<commit>^..<commit>` : idem anterior, incluyendo el primer commit

revert

revierte los cambios de un commit haciendo otro commit

- `git revert <commit>`
  - `-e` : permite editar el mensaje de commit (acción por defecto en línea de comando)
  - `--no-edit` : lo contrario
- `git revert <commit>..<commit>` : revierte desde el primero (no incluido) al último
  - `-n` : no realiza el commit ¿?
  - para incluir el primero, usar `<commit>^`

reset

- `git reset <commit>` : elimina los commits posteriores a <commit> y deja las modificaciones en el **working copy**
- `git reset HEAD~1` : elimina el último commit
- `git reset <commit>^` : elimina el commit seleccionado y posteriores

> **elimina commits posteriores y elimina completamente las modificaciones**
>
> git reset –hard 'commit'

rebase

> **reescritura de la historia**

- `git rebase <rama>`
  - `-i` : interactivo
- `git rebase -i HEAD~n` : rabase interactivo de los últimos **n** commits
- `git pull --rebase` : al recuperar del remoto, se actualiza la historia
- `git config --global pull.rebase true` : establecerlo por defecto

+ info

- http://cambrico.net/git-control-de-versiones/rebase-en-git

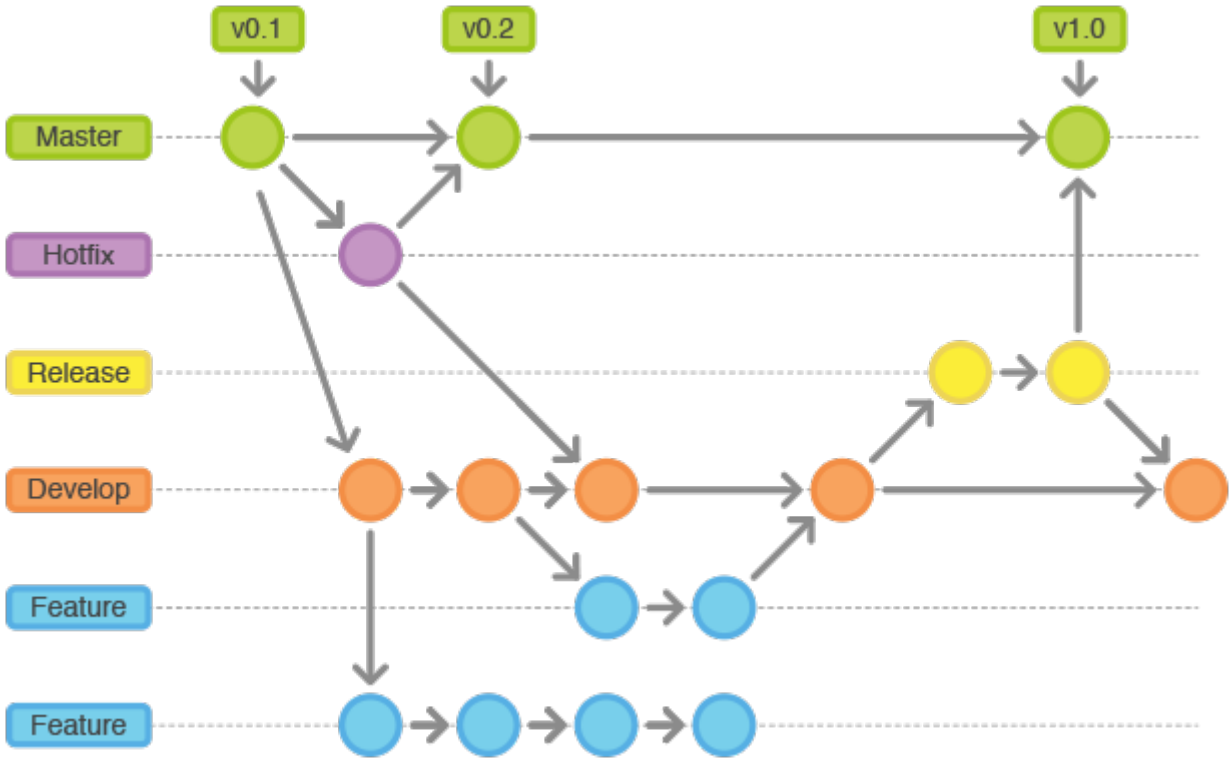
- https://git-scm.com/book/es/v1/Ramificaciones-en-Git-Reorganizando-el-trabajo-realizado
- https://www.solucionex.com/blog/git-merge-o-git-rebase

reflog

- `git reflog` : muestra las acciones realizadas en el repositorio
- `git merge <id_reflog>` : merge commits perdidos
- `git checkout -b <rama> <id_reflog>` crea rama con la información del pasado

FLOW

- patrón para el nombrado de ramas: **categoria/tipo-ID**
  - catetorias: feature, hotfix



[1)]

Daniel Sanchez Puig